Machine learning phase transitions in a scalable manner on classical and quantum processors

MARINA MARINKOVIC

ETH Zürich

marinama@ethz.ch

in collaboration with:

JOHN CORMICAN

Trinity College Dublin

The University of Dublin

ARTURO DE GIORGI

LMU Ludwig-Maximilians-Universität München

ETH zürich

Instituto de Física Teórica UAM-CSIC

Applications of ML in Lattice Gauge Theories

- Ever-increasing amounts of data, but also insufficiently efficient algorithms call for alternative approaches to conventional lattice QCD paradigm

- Areas of lattice applications so far:

See talk by G. Kanwar

- Generation of gauge configurations
  [Hacket et al. 2107.00734 ,Albergo et al. 2106.05934]
  [Del Debbio et al. 2105.12481]
- Taming signal-to-noise ratio
  [Yoon et al. 1807.05971]   [Nicoli et al. 2007.07115]

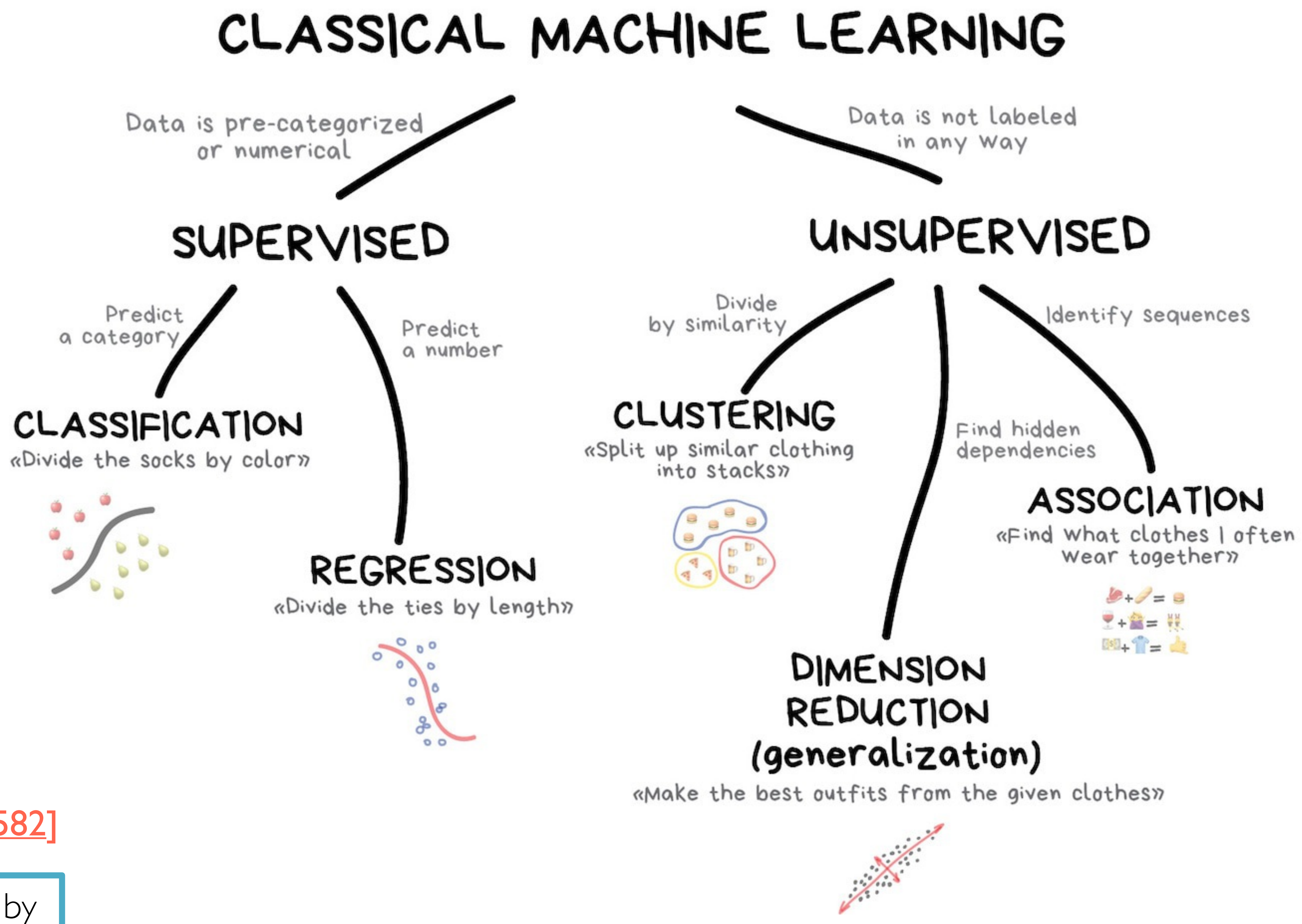
See talk by W. Detmold

- Reducing sign problem by contour deformation
  [Lawrence et al. 2101.05755, Detmold et al 2009.10971, Alexandru et al. 1709.01971, Mori et al. 1705.05605]

See talks by S. Wetzel, S. Paul N. Sale and S. Funai

- Phase separation, order parameters
  [Aarts et al. 2007.00355, Boyda et al 2009.10971, Wetzel et al. 1705.05582]
- …

For a recent overview, see plenary by Sebastien Racaniere @LATTICE 21



[Credit: https://vas3k.com/blog/machine_learning/]

Applications of ML in Lattice Gauge Theories

- Ever-increasing amounts of data, but also insufficiently efficient algorithms call for alternative approaches to conventional lattice QCD paradigm

- Areas of lattice applications so far:

See talk by G. Kanwar

- Generation of gauge configurations
  [Hackett et al. 2107.00734 ,Albergo et al. 2106.05934]
  [Del Debbio et al. 2105.12481]
- Taming signal-to-noise ratio
  [Yoon et al. 1807.05971] [Nicoli et al. 2007.07115]

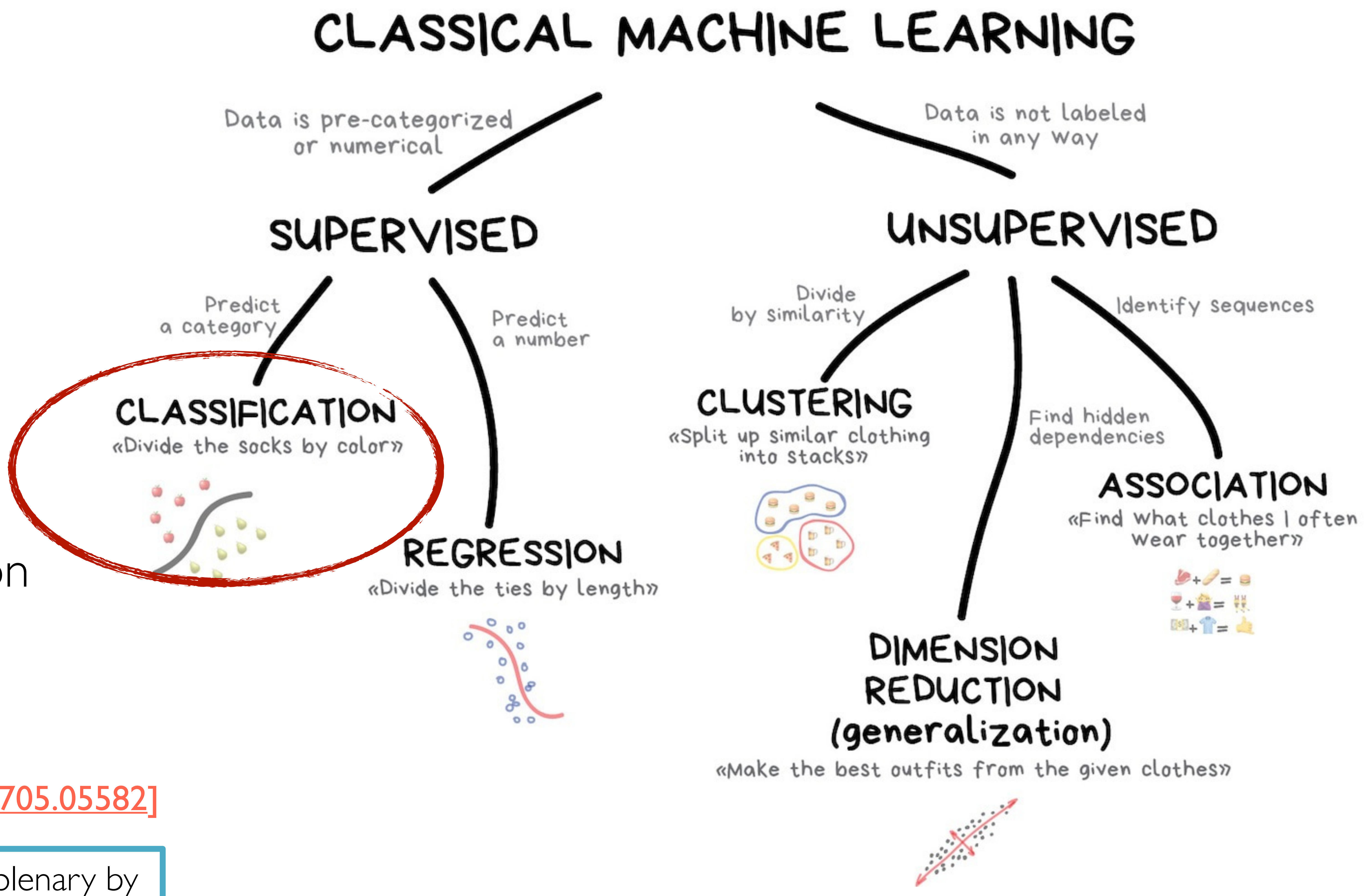
See talk by W. Detmold

- Reducing sign problem by contour deformation
  [Lawrence et al. 2101.05755, Detmold et al 2009.10971, Alexandru et al. 1709.01971, Mori et al. 1705.05605]

See talks by S. Wetzel, S. Paul N. Sale and S. Funai

- Phase separation, order parameters
  [Aarts et al. 2007.00355, Boyda et al 2009.10971, Wetzel et al. 1705.05582]
- …

For a recent overview, see plenary by Sebastien Racaniere @LATTICE 21



[Credit: https://vas3k.com/blog/machine_learning/]

Suport Vector Machine applications to phase separation

- Some examples of machine learning application in condensed matter and particle physics: Support Vector Machines [Carrasquilla&Melko, 1605.01735, Giannetti et al. 1812.06726, Woloshyn, 1905.08220, Liu et al. 1905.05125]

- Other learning approaches used for successful classification e.g. Restricted Boltzmann Machines, Convolutional Neural Networks, deep learning autoencoders etc. [Cosu et al. 1810.11503, Hu et al. 1704.00080, Alexandrou et al., 1903.03506]

- Phase separation in systems with fermion sign problem using CNN [Broecker et al, arXiv:1608.07848] and real time dynamics —-> See talk by J. Carrasquila

- SVM used for phase classification and determination of critical parameters in the Ising and Potts models, also ϕ^4 theory in 2d

Suport Vector Machine applications to phase separation

- Some examples of machine learning application in condensed matter and particle physics: Support Vector Machines [Carrasquilla&Melko, 1605.01735, Giannetti et al. 1812.06726, Woloshyn, 1905.08220, Liu et al. 1905.05125]
- Other learning approaches used for successful classification e.g. Restricted Boltzmann Machines, Convolutional Neural Networks, deep learning autoencoders etc. [Cosu et al. 1810.11503, Hu et al. 1704.00080, Alexandrou et al., 1903.03506]
- Phase separation in systems with fermion sign problem using CNN [Broecker et al, arXiv:1608.07848] and real time dynamics —-> See talk by J. Carrasquila
- SVM used for phase classification and determination of critical parameters in the Ising and Potts models, also ϕ^4 theory in 2d

SVM application to characterise phase transions

- Supervised SVM successful for phase classification in two dimensions

- For higher-dim: large training sets for accurate results; serial classical codes take very long

Option 1) Parallelize the code

Option 2) Speed-up the learning

SVM application to characterise phase transions

- Supervised SVM successful for phase classification in two dimensions

- For higher-dim: large training sets for accurate results; serial classical codes take very long

Option 1) Parallelize the code

- **a)** Resort to publicly available parallel libraries
- **b)** Physics problems: custom-made solutions

Option 2) Speed-up the learning

SVM application to characterise phase transions

- Supervised SVM successful for phase classification in two dimensions

- For higher-dim: large training sets for accurate results; serial classical codes take very long

Option 1) Parallelize the code

- **a)** Resort to publicly available parallel libraries
- **b)** Physics problems: custom-made solutions

Option 2) Speed-up the learning $\longrightarrow$ quantum kernels

SVM application to characterise phase transions

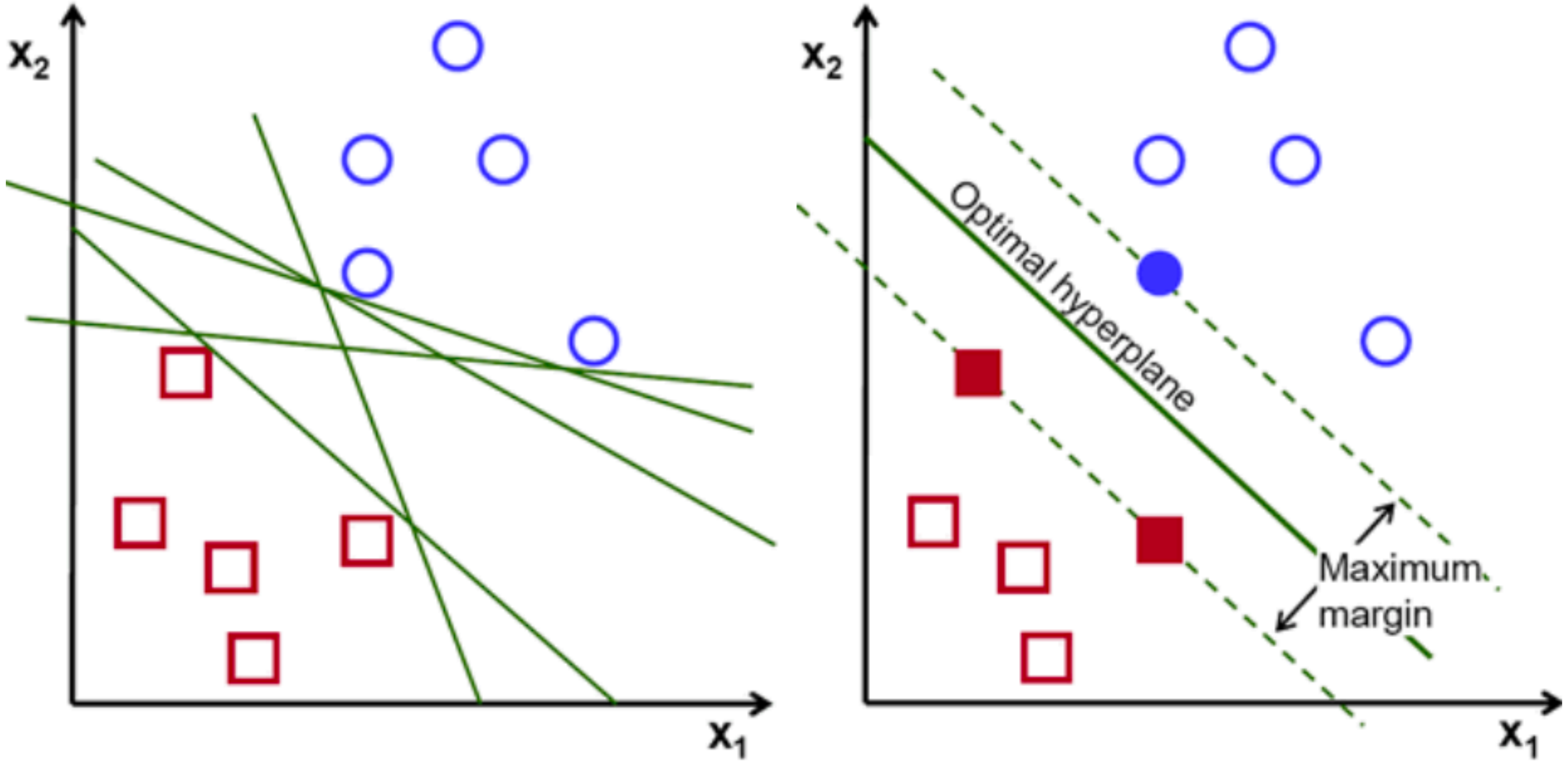
- Supervised SVM successful for phase classification in two dimensions

- For higher-dim: large training sets for accurate results; serial classical codes take very long

Option 1) Parallelize the code
- **a)** Resort to publicly available parallel libraries
- **b)** Physics problems: custom-made solutions

Option 2) Speed-up the learning $\longrightarrow$ quantum kernels

Support Vector Machine Algorithm



[Source: https://towardsdatascience.com/support-vector-machine-vs-logistic-regression-94cc2975433f]

- Linear SVM: training the construction of **d-1** dimensional hyperplane from **M** samples of **d** dimensional vectors, belonging to one of two classes
- Correctly separates the classes while maximizing the Euclidean distance between the hyperplane and nearest training vectors

Support Vector Machine Algorithm

- Data set: $\left\{(\overrightarrow{x_j}, y_i) : \overrightarrow{x_j} \in \mathbb{R}^d, y_j = \pm 1\right\}_{j=1...M}$

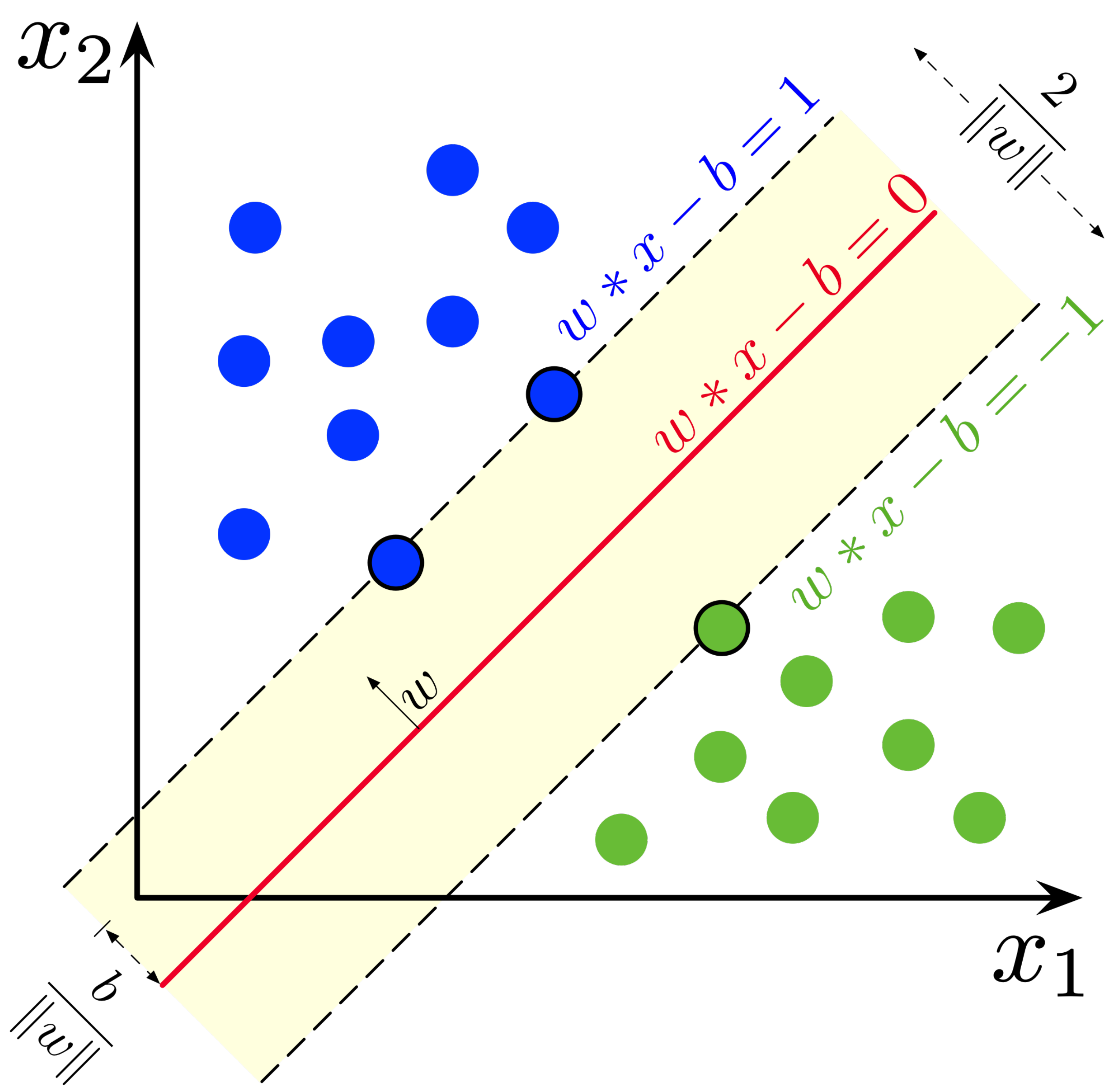
  - minimize $\frac{1}{2} | \overrightarrow{w} |^2$ subject to constraints $y_j(\overrightarrow{w} \cdot \overrightarrow{x_j} + b) \geq 1$

- Equivalent to a maximization problem:

$$L(\overrightarrow{\alpha}) = \sum_{j=1}^{M} y_j \alpha_j - \frac{1}{2} \sum_{j,k=1}^{M} \alpha_j \alpha_k (\overrightarrow{x_i} \cdot \overrightarrow{x_j})$$

- Get hyperplane parameters:

$$\overrightarrow{w} = \sum_{j=1}^{M} \alpha_j \overrightarrow{x_j}, \quad b = y_j - \overrightarrow{w} \cdot \overrightarrow{x_j} \text{ for } j \text{ where } \alpha_j \neq 0$$



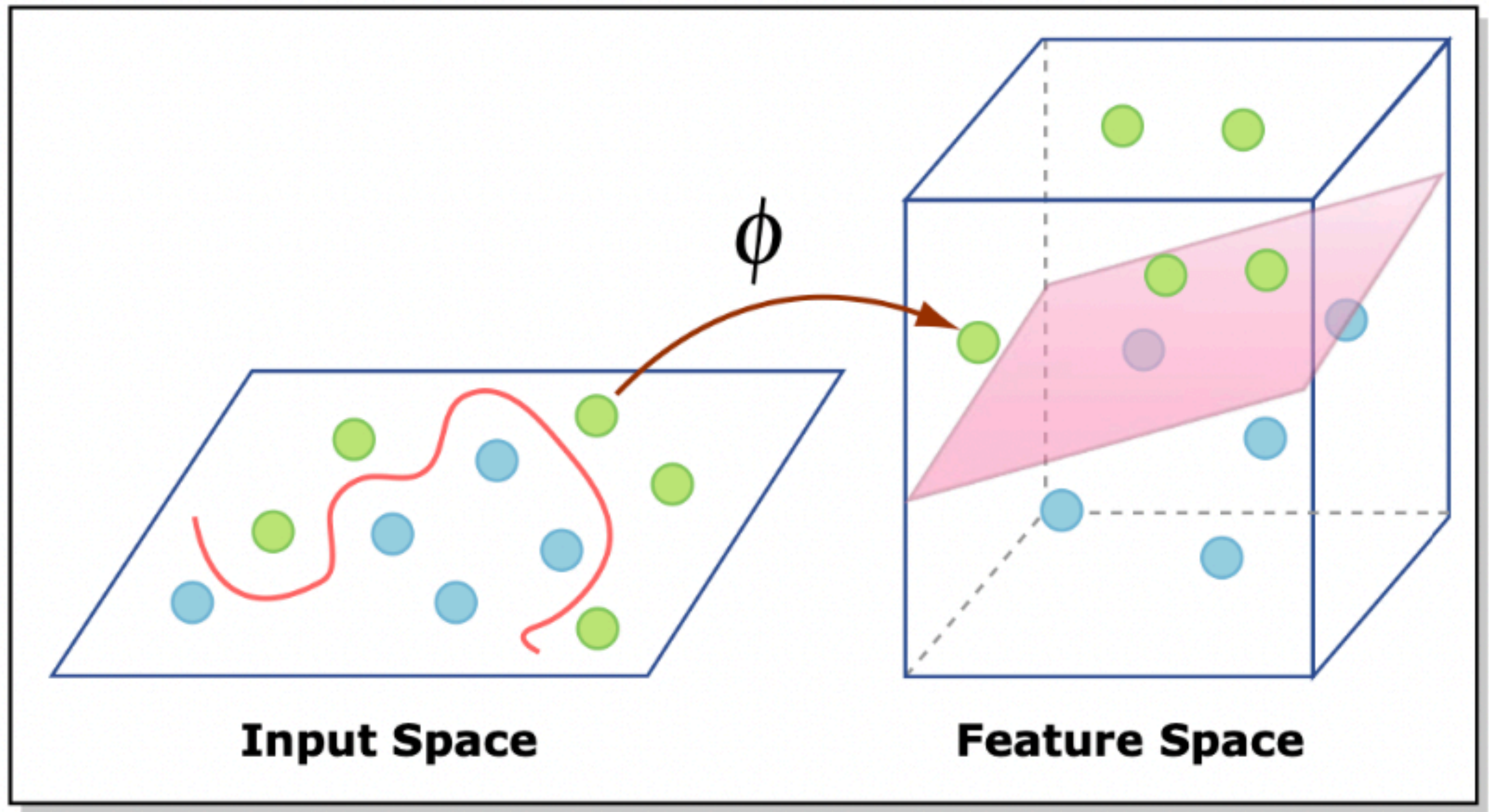
[Credit: https://commons.wikimedia.org/w/index.php?curid=73710028]

Support Vector Machine Algorithm

- Data sets that are not linearly separable:
  - tranformation of a non-linear problem into a linear one by moving to higher dim. space

Feature Map: $\phi : \mathbb{R}^d \mapsto \mathbb{R}^D; x_i \mapsto \phi(x_i) \equiv \phi_i$

Kernel: $k : \mathbb{R}^D \times \mathbb{R}^D \mapsto \mathbb{R}; \quad K_{ij} \equiv k(x_i, x_j) \equiv \phi_i \cdot \phi_j$



[Credit: MIT OpenCourseWare]

- Data enters in $L(\vec{\alpha})$ only through $(x_i \cdot x_j)$
  - knowledge of the kernel is sufficient:

$$L(\vec{\alpha}) = \sum_{j=1}^{M} y_j \alpha_j - \frac{1}{2} \sum_{j,k=1}^{M} \alpha_j K_{i,j} \alpha_k \qquad \Longrightarrow \qquad y(\vec{x}) = \mathrm{sign}\left(\sum_{j=1}^{M} \alpha_j \, k(\vec{x_j}, \vec{x}) + b \right)$$

SVM application to characterise phase transions

- Some examples of machine learning application in condensed matter and particle physics: Support Vector Machines [Carrasquilla&Melko, 1605.01735, Giannetti et al. 1812.06726, Woloshyn, 1905.08220, Liu et al. 1905.05125]

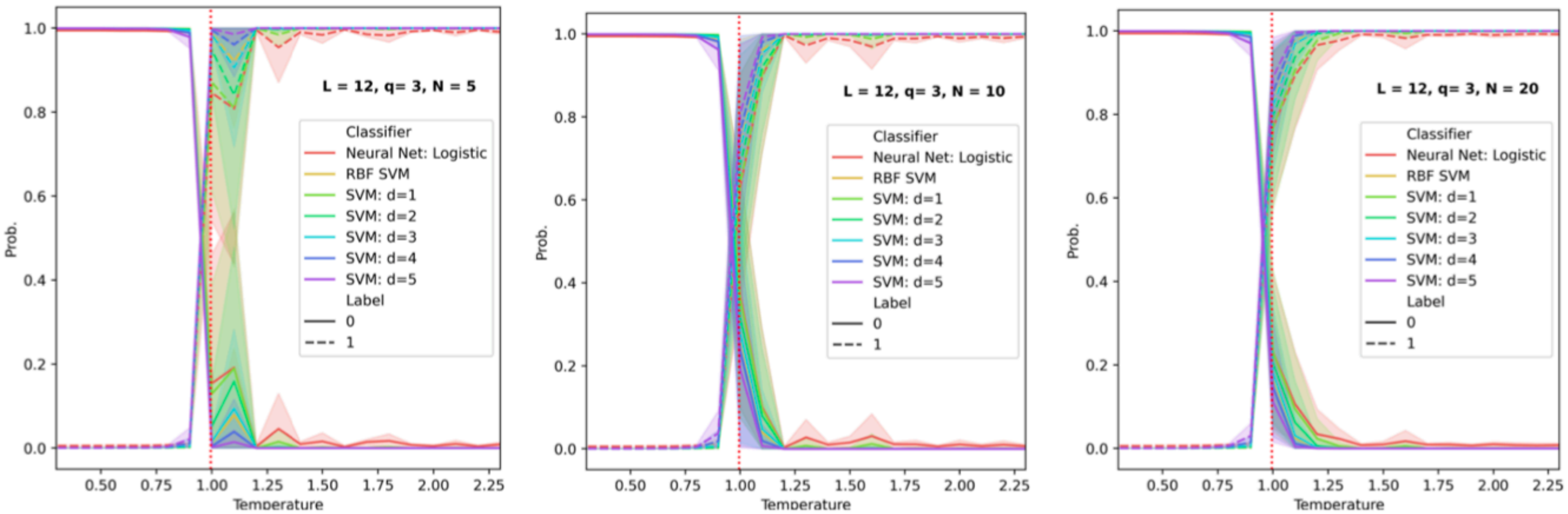
- Train machine learning algorithm (SVM and NNs) away from the critical region (training at T=0.5; T=5; testing $\Delta T = 0.1$)

- Few test configurations needed for reliable prediction of the phase transition parameters

- $$H = -J \sum_{\langle ij \rangle} \delta(\sigma_i, \sigma_j)$$

$$\sigma_i = 0,1,\ldots,q-1$$

- $q = 3$ Potts Model



[DeGiorgi, Marinkovic, in preparation]

SVM with Conjugate Gradient

- Maximization function: $F(\boldsymbol{\alpha}) = \boldsymbol{\alpha}^T - \frac{1}{2}\boldsymbol{\alpha}^T H \boldsymbol{\alpha}$
  $$H_{ij} = y_i \, (x_i \cdot x_j) \, y_j$$
  - subject to constraints: $0 \leq \boldsymbol{\alpha} \leq \boldsymbol{c}$
    $\boldsymbol{y}^T \boldsymbol{\alpha} = 0.$
- Karush-Kuhn Tucker (KKT) conditions for optimal α
- Sparse: $\alpha_i = 0$ for most training vectors
- Conjugate Gradient algorithm can be used to solve the optimization problem for α

Initialize $\boldsymbol{\alpha} = \mathbf{0}$, $\boldsymbol{r} = \mathbf{1}$;
Initialize P' for some initial active set sampling from both classes;
while $\boldsymbol{\beta} < \mathbf{0}$ *or* $P\hat{\boldsymbol{r}} \neq \mathbf{0}$ **do**
    Set $\hat{H}$,$\hat{P}$,$\hat{\boldsymbol{r}}_0$ and $\hat{\boldsymbol{\alpha}}$;
    Solve active set problem with CG;
    if *Boundary Conditions Violated* **then**
        Remove entry from active set;
    end
    $\boldsymbol{\alpha} = \boldsymbol{\alpha} + P'(P\hat{\boldsymbol{\alpha}})$;
    $\boldsymbol{r} = \boldsymbol{r} - H(P'(P\hat{\boldsymbol{\alpha}}))$;
    Calculate $\boldsymbol{\beta}$;
    Relax at most l active constraints with most negative β_i.;
    Update H,P,$\hat{\boldsymbol{y}}$;
end

[Wen et al., 2001, 10.1090/conm/323/05708]

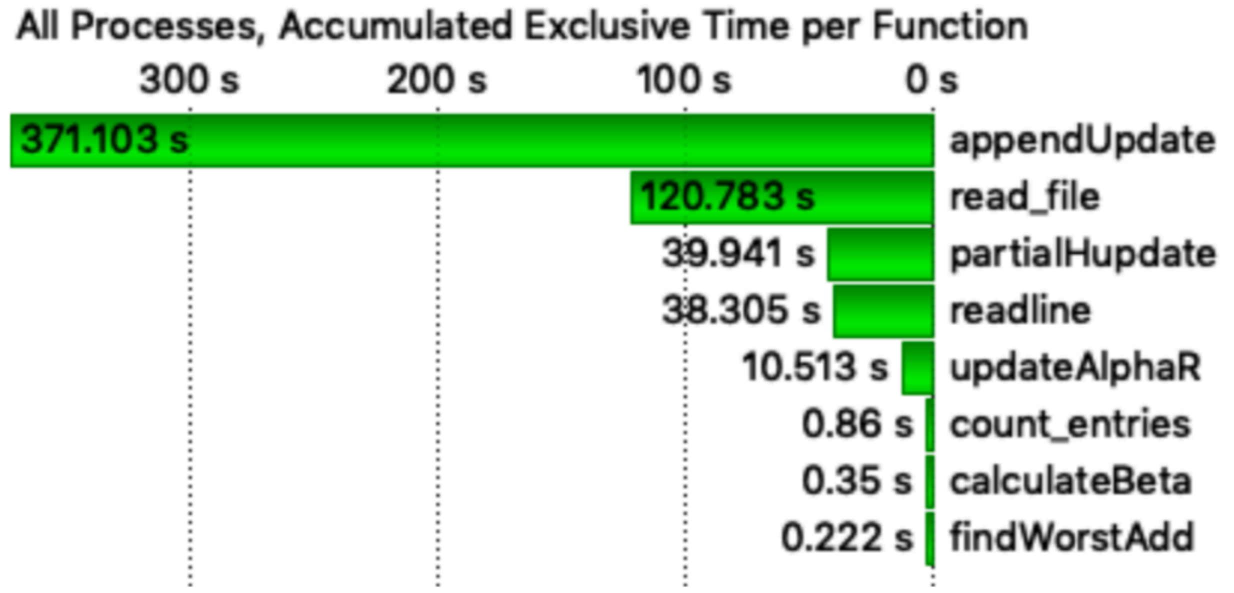
Parallel version of the SVM algorithm

- Multi-CPU Conjugate Gradient Implementation
- Data set divided across multiple nodes
- Hybrid approach: MPI + openMP
- One sided communications (RMA) beneficial
- Typical datasets with 200’000-400’000 entries

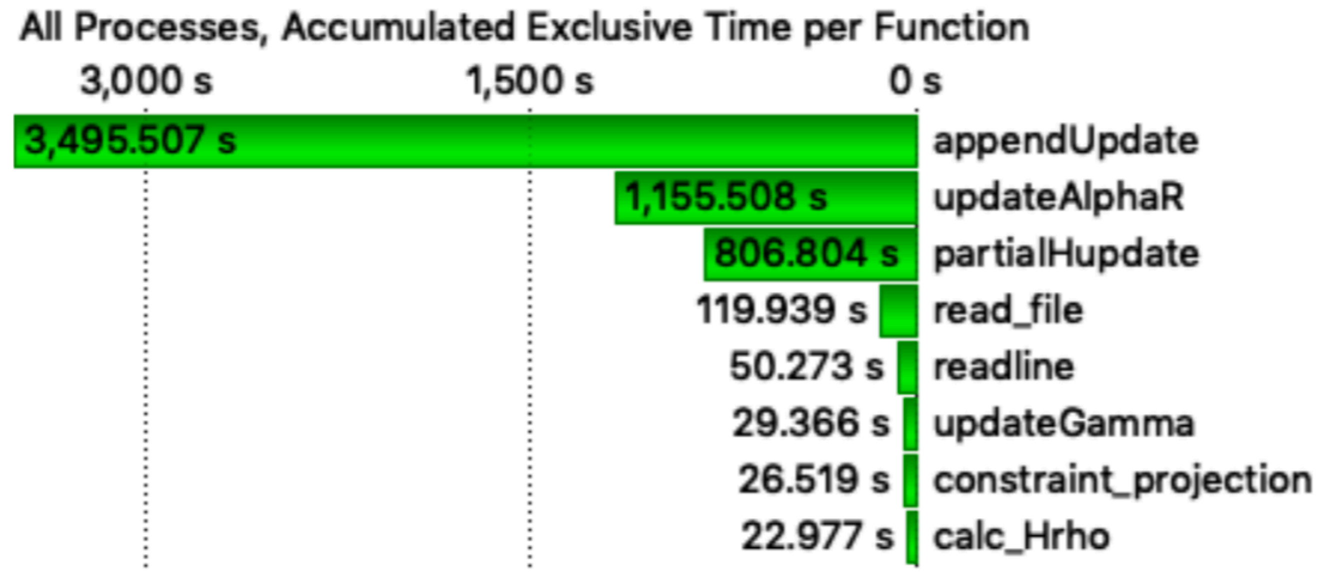
- Profiling and optimization of H (minimal storage and maximal efficiency)

$$H_{ij} = y_i\,(x_i \cdot x_j)\,y_j$$

Separable data:

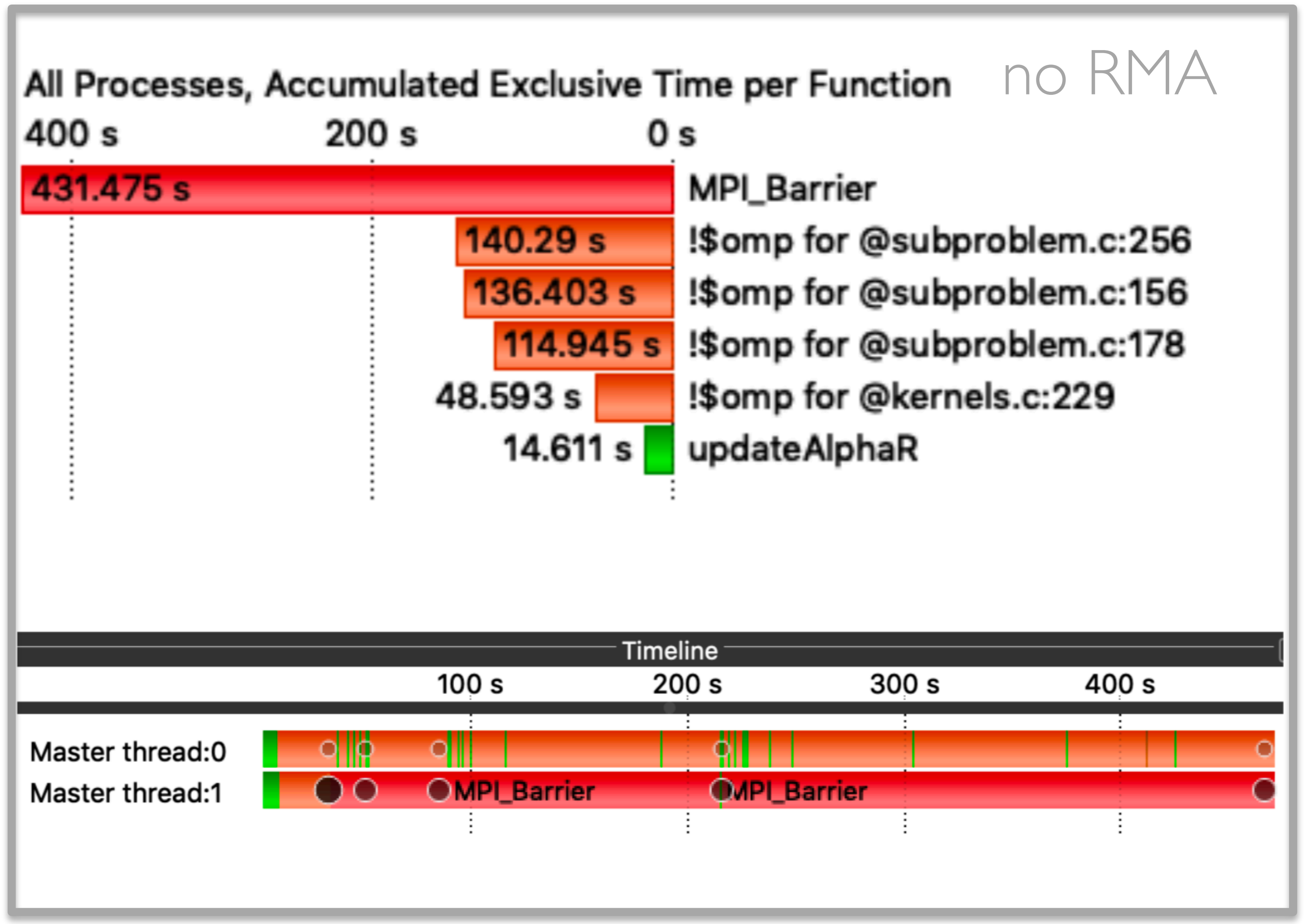


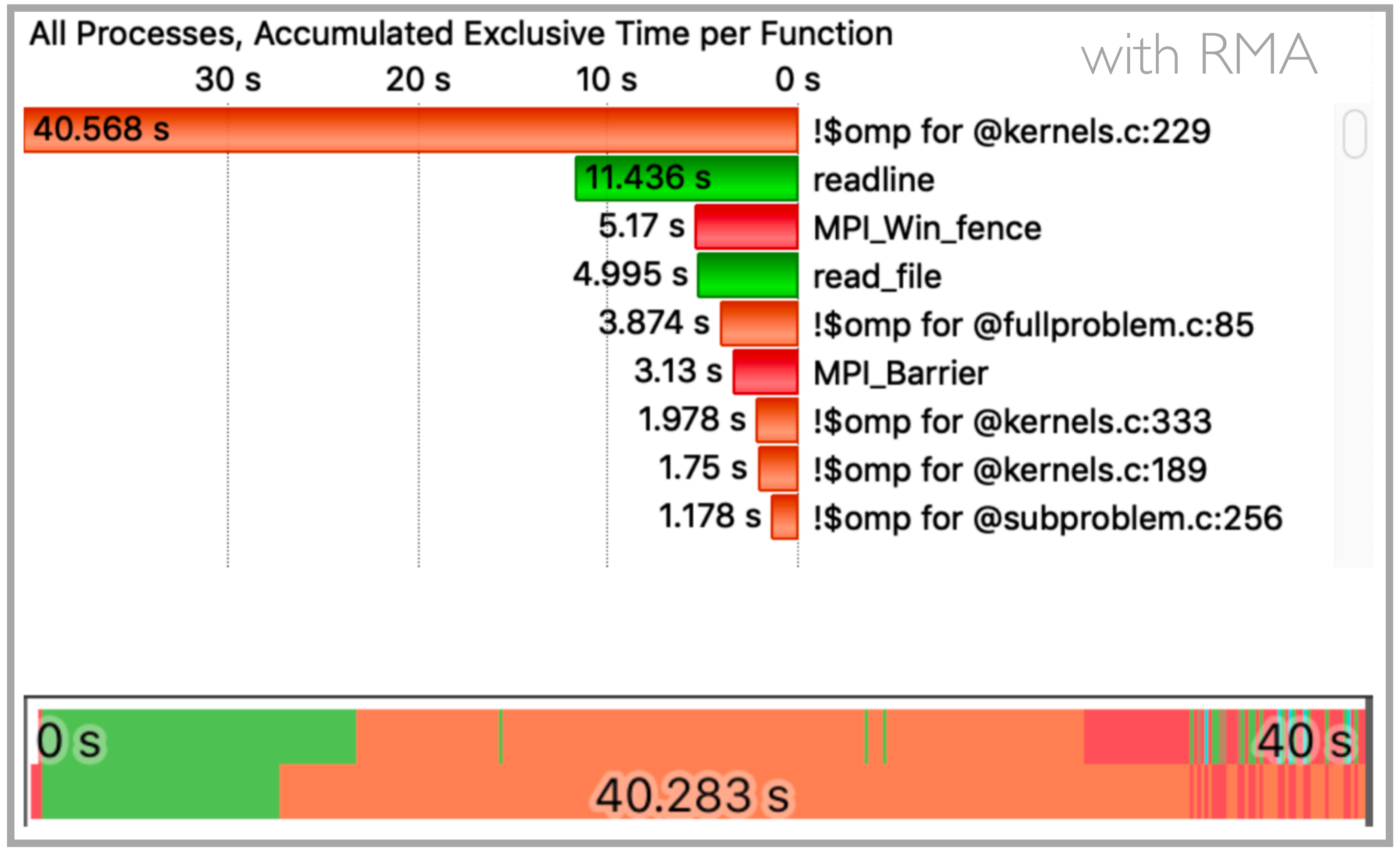
Non-separable data:



[J. Cormican, MSc theses, TCD, 2019]
[J. Cormican, MKM, PASC21]

Results on multiple CPUs

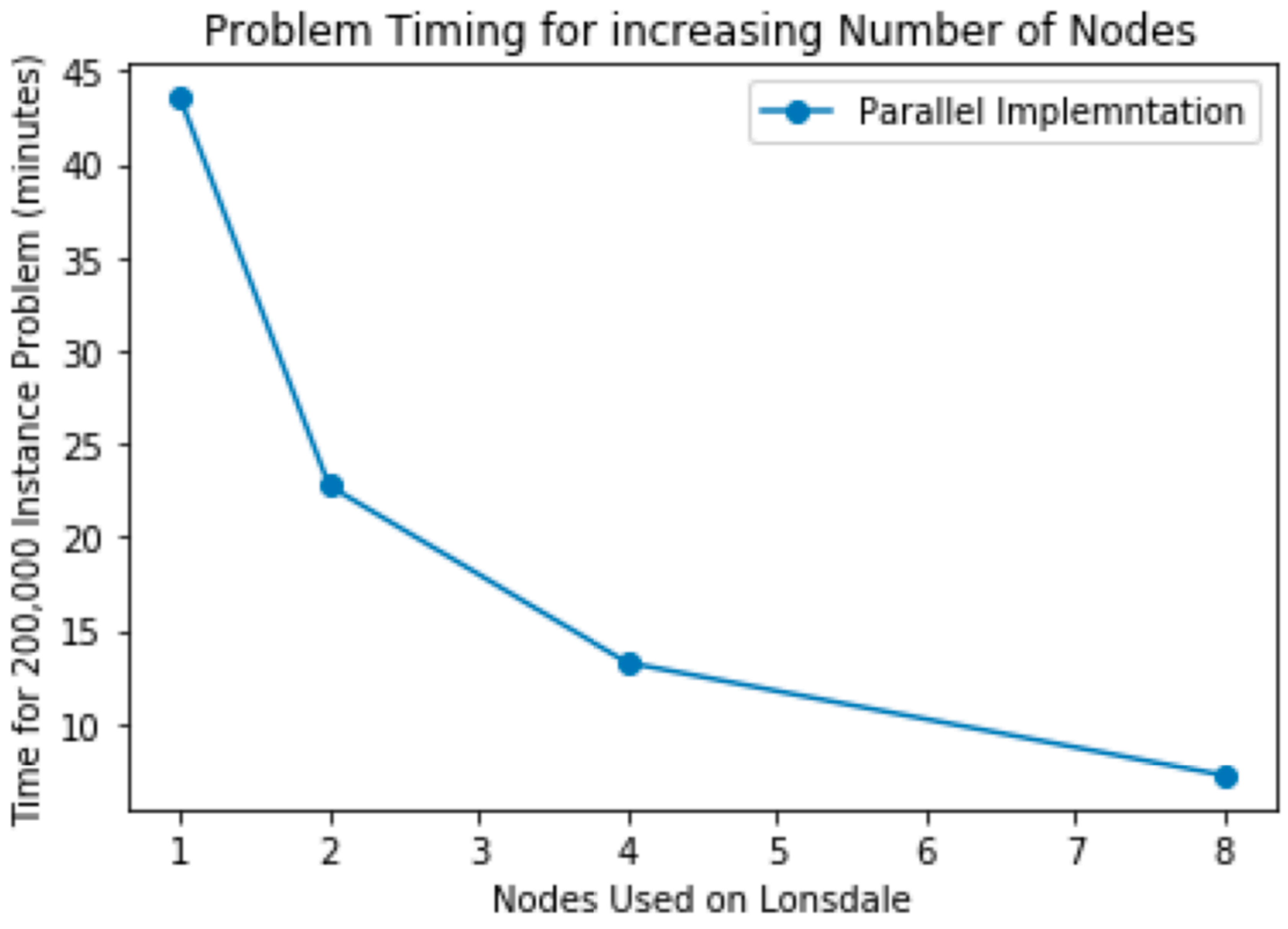


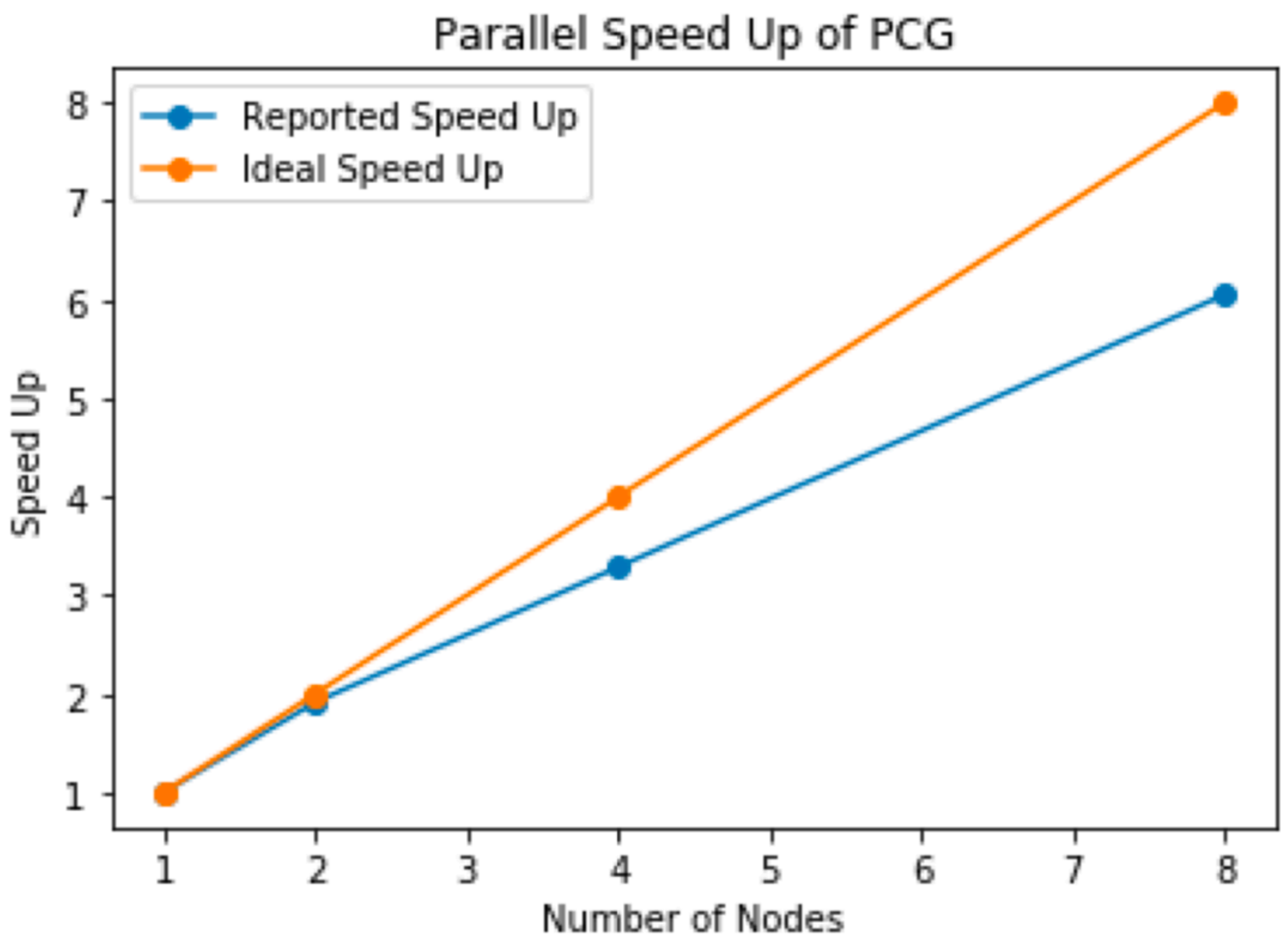


- Profiling with Vampir and Score-P)

- [J. Cormican, MSc theses, TCD, 2019; J. Cormican, MKM, PASC21]

Results on multiple CPUs





- Benchmarks on Lonsdale cluster @TCPHC)
- 1 node = 8 CPUs with 2.30GHz clock speed, no GPU
- [J. Cormican, MSc theses, TCD, 2019; J. Cormican, MKM, PASC21]

Quantum Suport Vector Machines

- Dimension of the feature space (N), and the size of the training set (M)
- Execution time for a given accuracy lengthy for big feature spaces and large statistics: $\Delta t \sim O(\mathrm{poly}(N, M))$

Quantum Suport Vector Machines

- Dimension of the feature space (N), and the size of the training set (M)
- Execution time for a given accuracy lengthy for big feature spaces and large statistics: $\Delta t \sim O(\mathrm{poly}(N, M))$

- Can quantum do it exponentially faster? In some cases yes.
- Quantum algorithm: $\Delta t \sim O(\log M\, N)$ [Rebentrost et al., Phys. Rev. Lett. 113 (2014)]

Quantum Suport Vector Machines

- Dimension of the feature space (N), and the size of the training set (M)
- Execution time for a given accuracy lengthy for big feature spaces and large statistics: $\Delta t \sim O(\mathrm{poly}(N, M))$

- Can quantum do it exponentially faster? In some cases yes.
- Quantum algorithm: $\Delta t \sim O(\log M\, N)$ [Rebentrost et al., Phys. Rev. Lett. 113 (2014)]

- **Quantum-Enhanced SVM:** the kernel is computed as quantum, but a classical SVM algorithm is followed
- [Schuld et al., Phys. Rev. A 101 (2020), Schuld et al. Phys. Rev. Lett. 122 (2019), Havlicek et al., Nature. vol. 567 (2019)]

Quantum-Enhanced SVM approach

[Schuld et al., Phys. Rev. A 101 (2020), Schuld et al. Phys. Rev. Lett. 122 (2019), Havlicek et al., Nature. vol. 567 (2019)]

Classical $\vec{x}$ $\xleftrightarrow{\text{how?}}$ Quantum $|\vec{x}\rangle$

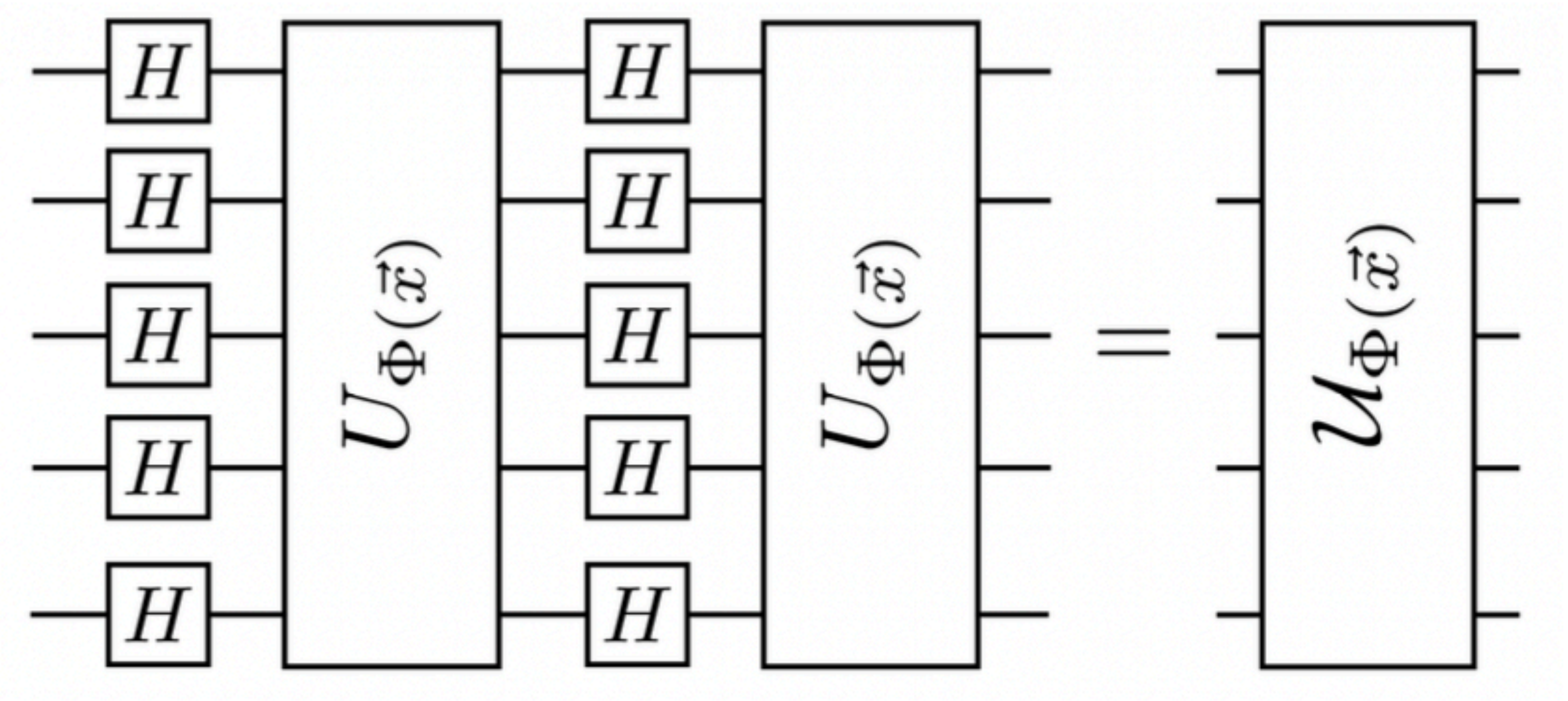
Map Classical Data to Quantum States

Quantum $|\vec{x}_n\rangle$, $|\vec{x}_m\rangle$ $\xleftrightarrow{\text{how?}}$ Classical $\vec{x}_n \cdot \vec{x}_m$, K_{nm}

Retrieve from States the Scalar Product / Kernel

Quantum-Enhanced SVM approach

- Define the **kernel** function as: $K(\vec{x},\vec{z}) = |\langle \Phi(\vec{x}) | \Phi(\vec{z}) \rangle|^2$



[Havlicek et al., Nature. vol. 567 (2019)]

- Define the **feature map** by the unitary circuit family:

$$U_{\Phi(\vec{x})} = \exp\left(i \sum_{S\subseteq[n]} \phi_S(\vec{x}) \prod_{i\in S} P_i \right) \qquad P_i \in \{I, X, Y, Z\} \quad S \in \{\tbinom{n}{k}\ combinations\ , k = 1 \ldots n\}$$

$$\mathcal{U}_\Phi(\vec{x}) = U_{\Phi(\vec{x})} H^{\otimes n} U_{\Phi(\vec{x})} H^{\otimes n} \qquad |\Phi(\boldsymbol{x})\rangle = \mathcal{U}_{\Phi(\boldsymbol{x})} |0\rangle^{\otimes n}$$

$$|\langle \Phi(\vec{x}) | \Phi(\vec{z}) \rangle|^2 = |\langle 0^n | \mathcal{U}^\dagger_{\Phi(\vec{x})} \mathcal{U}_{\Phi(\vec{z})} | 0^n \rangle|^2$$

Quantum-Enhanced SVM in Qiskit

qiskit 0.28.0
see release notes

Open-Source Quantum Development

Qiskit [quiss-kit] is an open-source SDK for working with quantum computers at the level of pulses, circuits, and application modules.

Get started

What can Qiskit do

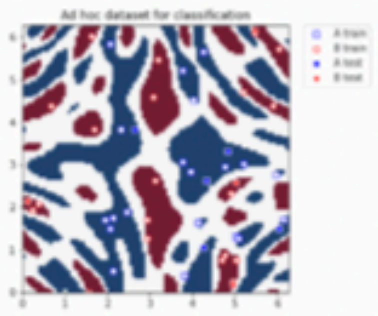
Qiskit accelerates the development of quantum app providing the complete set of tools needed for intera withquantum systems and simulators.

Qiskit

Machine Learning Tutorials

Quantum Neural Networks

Neural Network Classifier & Regressor



Quantum Kernel Machine Learning

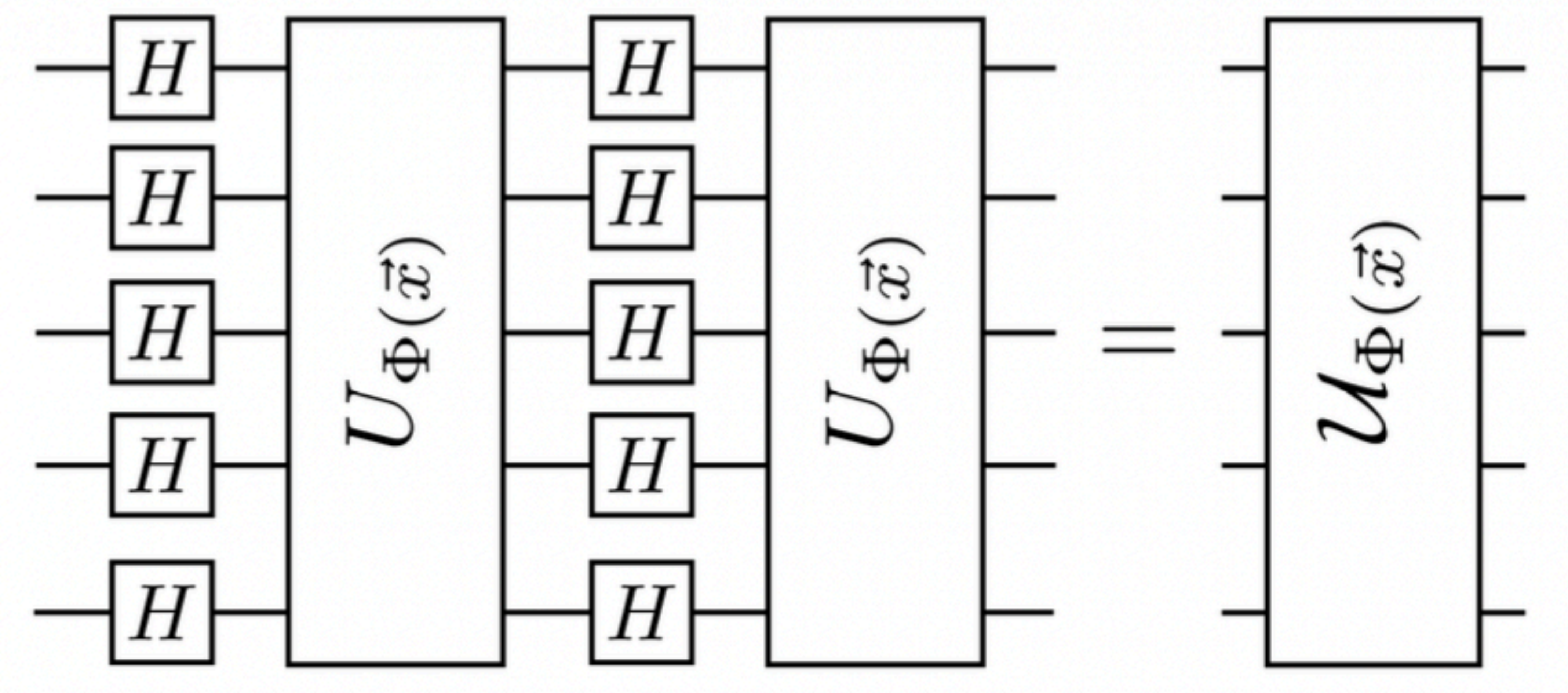
qGANs for Loading Random Distributions

Torch Connector and Hybrid QNNs

[Credit: https://qiskit.org/documentation]

- **Qiskit:** a **python** open-source software development kit for working with quantum computers at the level of circuits and algorithms
- QML packages available
- Execution on superconducting qubits via IBM Quantum Experience

Quantum-Enhanced SVM in Qiskit



[Havlicek et al., Nature. vol. 567 (2019)]

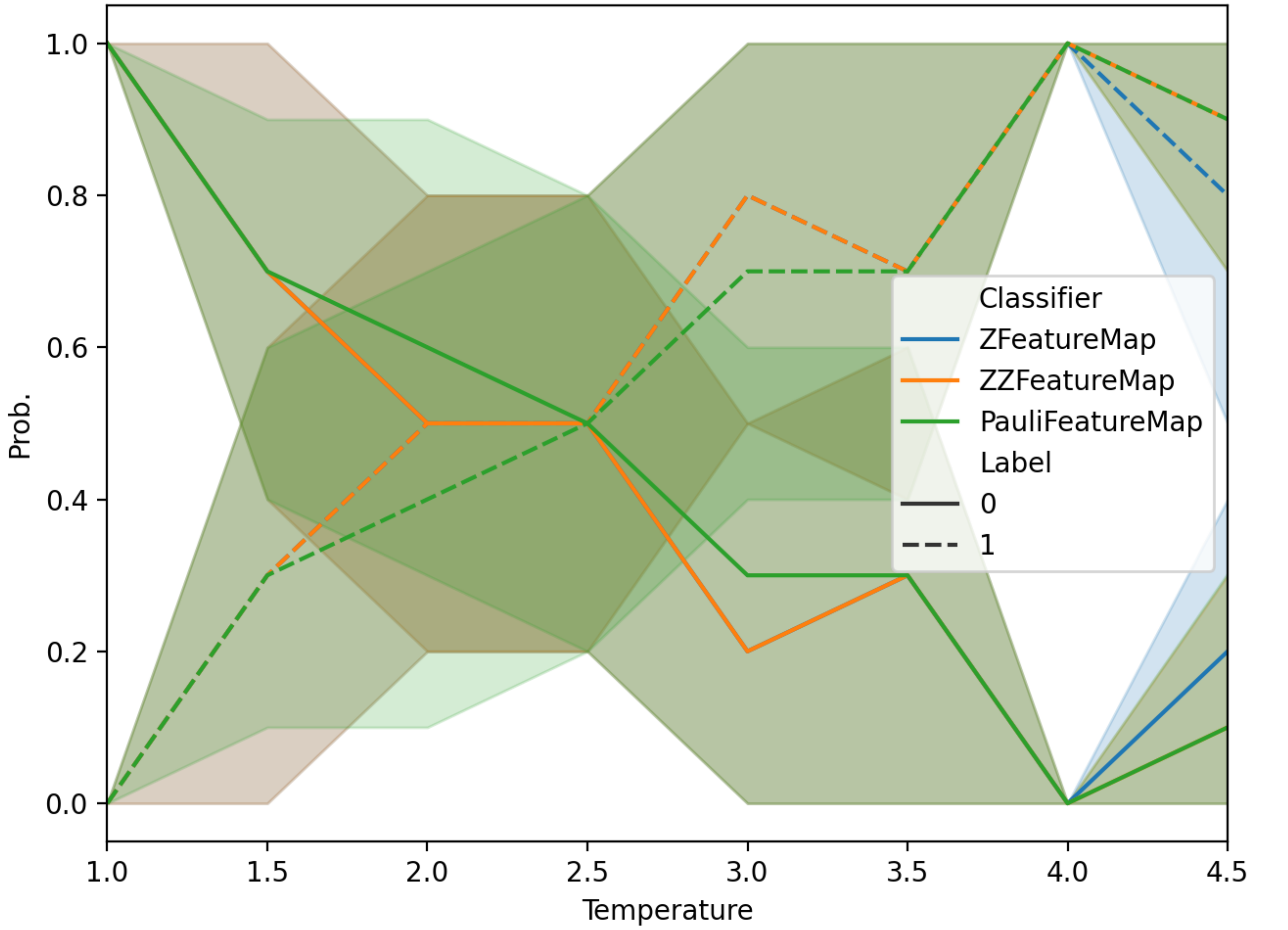
[https://qiskit.org/]

1. *FirstOrderExpansion* : $S \in \{0, 1, \ldots n-1\}$, so $\Phi_i(\boldsymbol{x}) = x_i$ — `ZFeatureMap` for the case $k = 1$, $P_0 = Z$
2. *SecondOrderExpansion*: $S \in \{0, 1, \ldots n-1, (0,1), (0,2) \ldots (n-2, n-1)\}$ and $\Phi_i(\boldsymbol{x}) = x_i$, $\Phi(\boldsymbol{x})_{ij} = (\pi - x_i)(\pi - x_j)$ — `ZZFeatureMap` for the case $k = 2$, $P_0 = Z$ and $P_{0,1} = ZZ$
3. *PauliZExpansion*: $S \in \{\binom{n}{k}$ *combinations*, $k = 1 \ldots n\}$ and $\Phi_S(\boldsymbol{x}) = x_i$ if k=1, $\Phi_S(\boldsymbol{x}) = \prod_S(\pi - x_j)$, $j \in S$otherwise

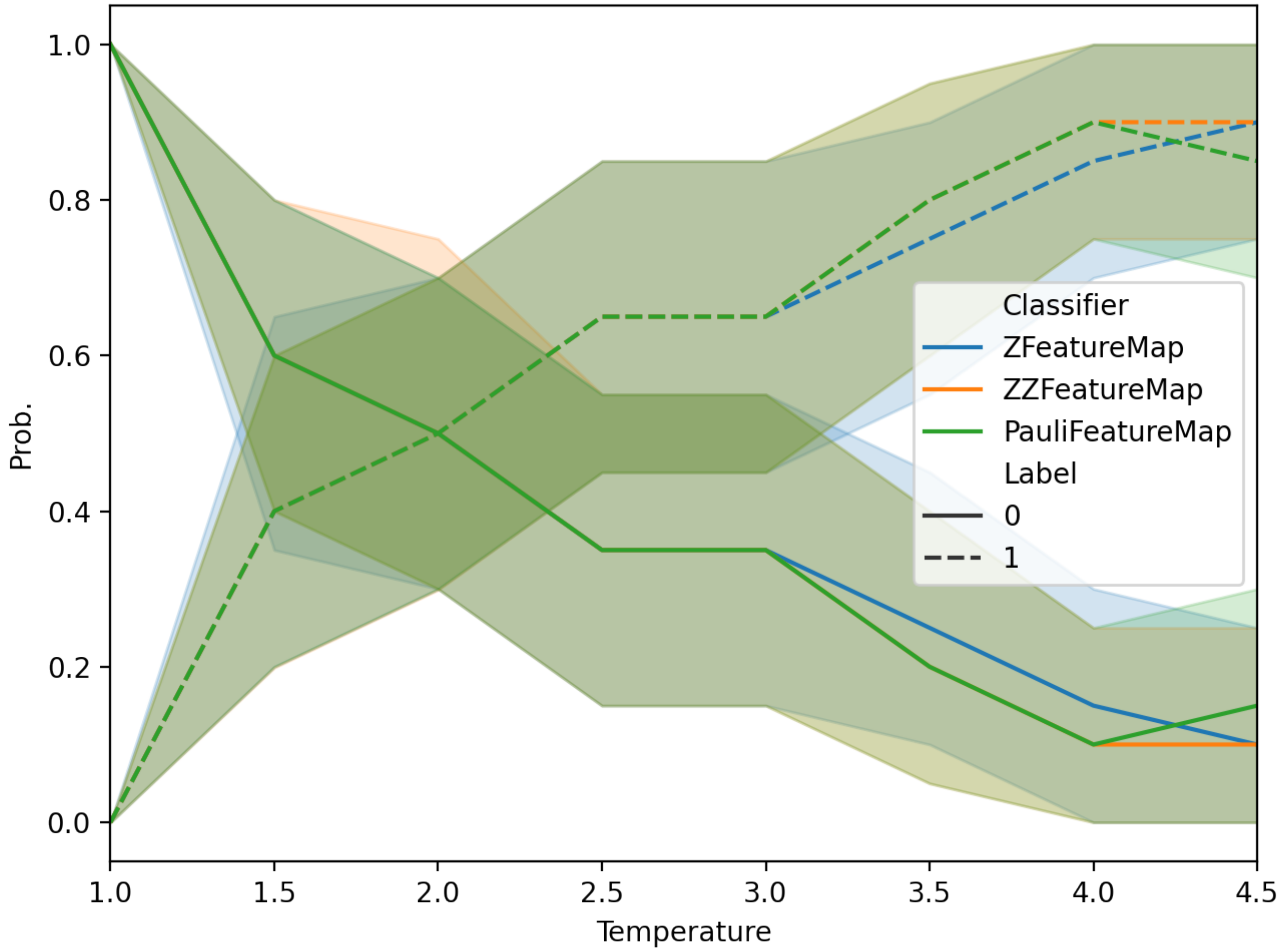
Phase Classification with Quantum ML

- 2D Ising Model ($T_c \approx 2.27$)
- 3x3 lattice, 32 samples for training at T=0.5 and 5; N test samples for each T in between ($\Delta T = 0.5$)

$N = 20$



$N = 80$



- Results obtained with IBM/Q machine ibmqx2 with 5 qubits

Phase Classification with Quantum ML

- 2D Ising Model ($T_c \approx 2.27$)
- 3x3 lattice, 32 samples for training at T=0.5 and 5; $N = 20/80$ test samples for each T in between ($\Delta T = 0.5$)

	Classifier	Score	Training Time [s]	Scoring Time [s]
0	ZFeatureMap	0.6750	46.124769	22.618253
1	ZZFeatureMap	0.7000	59.400390	46.003702
2	PauliFeatureMap	0.7125	136.765284	159.846363
3	Classical: Linear	0.7375	0.003784	0.004894

- Work in progress, results obtained with IBM/Q machine ibmqx2 with 5 qubits
- Slightly better score for quantum than for classical kernels
- Timing comparison not informative, in the quantum case includes waiting times on IBM/Q

Conclusions & Outlook

- Explore efficient execution of ML algorithms (starting from SVM) and apply it to learning critical parameters of the physical systems
- SVM used for phase classification and determination of critical parameters in the Ising, Potts Model and ϕ^4 in 2d
- Development towards d-dimensions in progress: among other developments faster classification needed
- For ϕ^4 neural networks more efficient than SVM, extend SVM-CG approach to parallelize generative networks
- Speed up the parallel implementation further by writing/importing and optimizing a GPU-friendly parallel-CG

Trinity College Dublin
The University of Dublin
LMU Ludwig-Maximilians-Universität München
ETH zürich

This work was supported by TCHPC (Research IT, Trinity College Dublin), Physik-IT at Ludwig Maximilian University of Munich and Euler cluster managed by the HPC team at ETH Zurich.

CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

Parts of this work are supported by a grant from the Swiss National Supercomputing Centre (CSCS) under project ID c21.

IBM Quantum
Qiskit

We acknowledge use of the IBM Q for this work. The views expressed are those of the authors and do not reflect the official policy or position of IBM or the IBM Q team.